date: July '94 #105

JOHN McQUILLEN INTERVIEW

Okay. Well, you're taking me into a pretty technical area. You know that.

Yeah. And I as a non-technical person must tape it. Yeah.

All right. Well, gosh, I haven't worked on this for more than 20 years. I'll see if I can remember it.

I know. It's been very tough, John, getting everyone's memories. So you're not alone.

Okay. So you want me to explain the spurious ack. The way the IMPs communicated was to send a packet from one IMP to the next and to receive an acknowledgement back and the storm forward protocol. IMPs would have several packets in queue and would send those packets off, and as acknowledgements came back, would release the packets from the queue. If an acknowledgement wasn't received in a certain amount of time, then the packet would be retransmitted. So it's a positive acknowledgement retransmission system. Things were working fine until we changed the software one time.

You and Crowther?

I changed the software one time. It was me.

And you were working at BB&N at the time?

Yeah. I worked at BB&N for ten years. I had the responsibility for writing all the IMP software and making all the releases into the network from something like 1972 to 1974. This problem came up in probably '71 or '72. The problem was that the IMP would get an ack back and there wasn't a packet for it. It really shouldn't have happened. The system was very simple. You send the packets out and they all came back. I had developed a kind of a clever way of having each packet that was coming back in the reverse direction carry acknowledgements in it, and in fact, it could carry several acknowledgements in it very inexpensively in (what we first had.) So we thought that was very clever and we thought that the system was kind of simple and fool proof. We had the spurious ack problem that was very, very unlikely. It took testing the system extensively and intensively to make it fail at all. I had to build fairly elaborate networks in the laboratory to make it fail and I had to run it for hours.

You had to simulate failure.

I had to simulate a real network with a lot of real traffic to get it to fail. Then it would fail once and I'd be left with a sort of bunch of computer corpses and I couldn't figure out what the problem was. It doesn't reflect that well on me, because it took me days and days and ultimately weeks of really intensive effort of being there night and day, having only a few failures to work with. Until I finally -- I kept tracking it down. And finally I discovered what the problem was. The problem was an interaction between the hardware and the software. You have to imagine that as packets are being sent out from the IMP, they're going out from memory, of course, and as packets are coming in, they're coming into memory, and as packets are being released from the queue when they're acknowledged, they're freed, and are put into a pre-place in memory.

The problem was that there was an interaction between the packets flowing in and out on the telephone lines and the time that the computer was using the buffers in memory. I'm not sure if I can explain enough of this to you. But the problem was that there was a race condition between when the computer freed the buffer and when the telephone line would start to fill it with data or start to empty it with data. I believe that the problem was this. That the packets were put on the queue to be sent and if they weren't acknowledged, they were sent again after a certain amount of time. I thought I'd been clever in deciding that if the IMP had nothing else to do, it could keep sending some of the old packets. The problem came about that if the IMP was sending one of these old packets and an acknowledgement came in for it, the IMP would then free that packet -- it's memory to be reused. If while the packet was still being transmitted by the telephone line, the memory that the packet was sitting in was then reused by another packet, it would get partially over written. So you have to imagine that the telephone line has already gotten this piece of memory that it's starting to send out from that piece of memory, and now if the computer gives this memory to somebody else and they start writing into it, what will actually flow out into the telephone line is half of one packet and half of another packet. Garbage. This only happened in the kind of race condition that the package was starting to be transmitted and it was acknowledged and if the memory was reused all within a few milliseconds of each other, so that the packet would get corrupted. It manifested itself in a couple of different ways. But the main way was that it would appear to be carrying acknowledgements that were meaningless, because some other packet had over written things on the IMP.

So the cure for that was -- there were multiple cures for it. One was just interlock things so that we didn't use the same memory for multiple things at the same time.

We didn't understand that we were, but we changed that. But more generally, we then put in a lot of integrity checks in the IMP to make sure that the packets were correct.

An interesting effect thing here is -- I don't know about interesting, but sort of technically interesting effect is that if you have this patch of memory that you've said, *"Okay, send out this packet."* You've learned about check (sums) and how these things are properly check summed. Okay. So there's a check sum of the bits that are flowing out of memory onto the wire. Now the bits that were flowing out of this memory were sent over the line correctly. So the check sum was okay. It's that we were sending the wrong bits. So we ended up putting in a second level of check sum, which was a check sum in software on the intended contents from memory. So we have one check sum to make sure there isn't a telephony error in transmission. Then we had a second check sum to make sure that there wasn't an integrity error in hardware or software in the IMP. That turned out to be very useful later for catching all kinds of hardware malfunctions and other things.

Who were you working with?

I was working with all of these guys. Will Crowther and Dave Walden and others. At this point, Will and Dave had stopped programming the IMP and I was programming it.

Were you actually at BB&N when the IMP was built?

I joined in the middle of '71.

Oh, okay. So it was after it had been installed.

It was after the first hardware and software was out there, and then, you know, it got entirely wrinkled multiple times. Right. So I wasn't part of the team that developed it in '68 and '69. But I was the one who developed the level two and level three protocols and the routing algorithms which were actually used for most of the 1970's.

Is this stuff that you had studied in college?

Yeah. I have my undergraduate and Masters and PhD degrees from Harvard in Computer Science. I got my undergraduate degree at Harvard in 1970 and as a grad student, I connected Harvard's PDP-1 to the Arpanet.

Really?

Ben Barker was at Harvard also. You've interviewed him?

Uh-huh.

And I was introduced to the Arpanet and IMPs and BBN by Ben. As a grad student, I built the hardware and software to connect this computer to the Arpanet and then I decided to go to BBN after I got my master's degree, and then over the next three years, from '71 to '74, I developed a whole new suite of protocols for the IMP. Basically, I rewrote the IMP.

Oh, really?

And then I wrote the new routing algorithm and I did my PhD work at Harvard on that part time. That is, I was working full time at BBN and I did my thesis part time.

What is the story of everything getting routed to Harvard at some point?

Yeah.

That's another story.

That's another story and it's kind of related to this. Now that we've talked about packets and acks, imagine it as a special kind of a packet which just says, *"From this IMP, how far away are all the other IMPs,"* and this says, *"Okay, for number one and two and three and four,"* you know, *"I'm six away from this and four away from that."*

And it chooses the shortest route.

Once all the IMPs have all this information, then they can all choose the best route.

Does this data constantly update itself?

Yeah.

It has to.

Right. And this is the stuff that I changed. But before I explain what I changed, I'll

just explain briefly what happened here. It's closely related to this problem. We thought that these routing tables were safe from errors and corruption because they have this hardware check sum -- any errors introduced by telephony problems. But what happens if a computer's memory fails, so that every now and again once it just reads out all zeros? Okay? I mean, you know, computers fail in all different ways.

The IMP at Harvard had a faulty memory buzz. So that every so often, when you went to read a memory location, you'd get a zero. In fact, you'd get all zeros. So it would construct a routing packet here that was a perfectly legitimate, syntactically correct routing packet with all zeros. Then it would go out over the network nicely check summed. All the other IMPs would get it and they would say, *"Boy, does Harvard have a really great route to all these other place."* And they'd just send all the traffic there and it became a black hole, because all the traffic went there, and like a black hole, we couldn't even get information out. See, once all the traffic is going there, even the kind of network management and control traffic, which we were using to remotely diagnose and debug, gets sucked into the gravitational orbit. So it's just like a black hole where you can't see it or observe it. You just know that something bad has happened.

Eventually, we had to kind of (cauterize) the network, cutting off that part of the network and then rebuilding it. Then we learned our lesson. As I say, these two stories are somewhat related. That one by one, we had to fix all the problems that should never even happen. You know, so the IMP was a very good lesson in a couple of things. One is truly distributed computations, so that in this first example of the spurious ack, we have two computers that are working on the same problem at the same time. Did this packet get through properly? If they're doing that, you have to worry about all the potential sequencing and simultaneity of the different things that they might do.

Then you have the routing problem, which is you might have 50 computers all working on the same problem at the same time, which is, you know, what's the best way to get to Utah? And if any one of them has a failure -- a heart rate failure -- you just have to make sure that the calculation proceeds. You just know that the computer's are going to have lightening storms, and power failures, and software bugs, and hardware bugs, and the janitor's going to trip the power cord, and just anything you can think of could happen. So we had to completely bullet-proof this distributed calculation so that it would keep working in the face of, as I say, quote *"impossible problems."* And we actually did that over a period of a couple of years and there were a few other funny, hah-hah, episodes where the whole network crashed because of problems like this. Then we finally got it completely licked.

So this caused the network to crash when this happened, because everything -- or did it?

Basically, yes.

What year was this?

Oh, this was very early. '73 or something.

So there were very few nodes.

Yes. And also, in this whole period of many years that I was involved in it, there were only two or three -- there were three examples of problems like this, and they took the network down for a matter of minutes or an hour until we just could basically cut it off and put it back together again. Which is a very good record, because it was already working perfectly fine except for really pathological conditions. Usually you'd expect if a computer is reading out all zeros that it'll just stop, because it can't read its own instructions or anything. So we're into this very, very intermediate case where it has a momentary amnesia, but then it keeps going, and that's not supposed to happen. You know, a computer's supposed to work or not work. I wish I could remember the other two, because they're all kind of beautiful examples of this sort of thing. Let me see if I can remember.

Maybe Walden might remember or Alec.

He might. But in any event, there were a couple of others of this type where the problem was that we needed to basically guarantee that no matter what the other computer said that was nonsense, we had to keep going.

When you say you basically rewrote the IMP, what does that entail?

Well, what it entails is that I made a completely new IMP program and I installed it into all the computers in the network.

That's a huge job.

50 times over a period of two years. I did it every two weeks.

You had to go each site?

No, no. I did it all remotely here. So Tuesday morning at 6:00 every other week,

I would bicycle into BBN and I would load up the first computer with my new software and then I would propagate it out. And this is a whole art form in itself, because if you think of the network, which in the early days, I don't know when I was starting it might have had 20 or 30 nodes. But by the end of a couple of years later, I'm sure it had 50 or 60 nodes. Here we are in Cambridge (hoof) on one side, you know, so this is all running old. And then you put new in here and then you put new in here and new in here. Well, of course, you have to have new work with new and you have to have old work with old, and you have to have old work with new, all at the same time. Because otherwise you can't then send the new program down here. So this is what we call backwards compatibility and forwards compatibility, and every time I had a change to make, if you think about a new change in a format of a message, you know, the message is going to start with this kind of indication instead of that kind. It'll start with acknowledgements instead of something else. I'd have to construct one or potentially more releases of software staged over a couple of hours or sometimes I'd do kind of one release and then I'd come back two weeks later and do another release. Every now and again, I'd put a release out at 6:00 and I'd get out to here and it would start not to work too well. We'd have to pull it back. So the releases had to be reversible so you could go back and put old in here and old in here. This whole things was really a lot of fun. This was good stuff. To be able to do all this and to do it all from Cambridge and everybody else would get in, in the morning, and the network would be fine and they wouldn't know whether it was the old software or the new software.

So when I say that I was writing it, what I mean is that Will and Dave and the others had written the whole thing in 1969, and it was up in 1970 and '71 when I got there, and the problems with it were clear. There were problems with the way the acknowledgements worked. There were problems with the way reassembly happened. There were problems with routing. And after I served a short apprenticeship rebuilding their Network Control Center -- or building their Network Control Center -- I built the first Network Control Center.

Oh, you did?

Yeah. So that was actually a key thing. So we had this Network Control Center over here. So I built that and then they sort of gave me the job of, *"Okay, well, why don't you start taking on the assignment of building all these new protocols?"* And so Crowther and I think some other people to a degree were involved in designing some of the new ideas and then I implemented them. And then I also began to do some of the design, and by the time this two or three year period was over and I was doing the routing, I was actually doing a lot of the design also. I literally wrote the

program, compiled the program. I had a terminal at home. Terminal in my office. Took hours and hours to compile this stuff with really antique equipment.

What was the equipment?

We were using a big old PDP-1 that BBN at used at Mass General Hospital. It was a timesharing system. This was the first timesharing system ever built. Had a model three teletype. So I'd go home and I'd set up these programs to compile and they'd run for hours or overnight, and then it would print out in the morning whether it completed properly or not, and then I'd compile the next piece and the next piece, and over a week or so, I'd get it altogether and I'd go into the test lab where I'd have three or four IMPs and I'd make sure it mostly worked, and then Tuesday morning, I'd start releasing it onto the network. In this way, I rebuilt all the way the acknowledgements worked. That's the level two stuff, from one IMP to another. All the end to end stuff, which is how the source IMP talks to the destination IMP is another set of messaging procedures. For that, I rebuilt all of that. And then I rebuilt the way the routing tables work and the whole routing algorithm. It took a lot of releases. A lot of _____ changes.

And was the end result substantially different from the original?

Well, I don't know how to answer that. I guess, there were critical design flaws in the first one that would have meant that it was going to be very difficult, if not impossible, to build a large operational network.

To grow the network.

To grow the network and to put operational traffic on it.

Do you think that was something sort of intrinsic in the nine-month scheduling realm? That things had to be done pretty quickly when they were first --

Yes. But also I think it's something intrinsic in just doing something for the first time.

This was all brand new territory.

It's totally, totally, totally brand new territory. I mean, not only was this -- I mean, it was brand new in every way. I don't mean it as a criticism at all that I had to redo all this. I mean, the people who did it knew that it had to be redone. They

were the ones who knew that. I think that it was quite miraculous to get a dynamic, adaptive, routing algorithm that ran as a distributive calculation running in the first place. Then you got it up on a network with a lot of nodes and then you could appreciate its strengths and its limitations. Likewise, with the acknowledgements. You know, we had a positive acknowledgement retransmission system. Probably the first one that was ever done. I mean, everything about this was first. It was the first time that anybody had ever built a distributed network of computers. So the acknowledgement system was the first thing of its kind and the host interface to computers was the first that had ever been built and it had to be invented. The routing thing -- it was the first time that had ever been done. So it's quite natural that a year or two later everyone said, *"Okay, well, now we know the way to really do this."*

And then, of course, that's been the story ever since. Every few years since, we've gone back and improved things. Sometimes when the hardware gets better. Sometimes when things have gotten bigger or faster. Sometimes when the use of the network changes. But this first one was going from a paper idea to a research prototype, which was really what the first IMP was, to then having a whole research network, which is what they did in '69, to then what I did was still a research network. What I was doing was still R&D, but it got to be a pretty big research network and it was starting to be used by real people. So it got into that grey area.

Did the growth surprise you?

The growth in those days didn't surprise me, because it was all a closed world then. These were all Arpa funded sites and Arpa's hand was on the steering wheel of all of this. They were giving us money to develop the technology and they were giving the sites money to use the technology.

That's right. So they held both.

Yeah. And they made the case before Congress that it was a savings of money to build this network, rather than giving more computers to each site. So they could kind of make the argument be true. The growth later, of course, I think it surprised everybody.

... ?

Yeah, I'm sure Vint Cerf has told you. Even though he probably had very high hopes for all of this, the growth has just blown everybody away. You know, when

we were doing this at the time, when there 9 computers and 15 computers and 30 computers and 50 computers on the network, I knew all the computers on the network.

You did?

Of course. Because I was releasing the software to all of them all of the time. I knew where they were and what their numbers were and who was there and I knew them all by name. To go from 50 to two million computers on the network is really amazing.

Still what astounds me is it's still basically the structure that you guys put in. I mean, the foundations of what you guys put in place is basically valid today.

That's right.

Is that true statement?

Yeah, that's right. It's still basically valid today. In fact, the routing algorithm work that I did is still -- the routing algorithm that I was replacing in the mid-70's is still around in some other implementations, and the one I replaced it with is definitely still around. Maybe I should explain very briefly about the routing stuff. Would that be helpful?

Uh-huh.

There's two kinds of routing algorithms. The one that I did is a generic one called (Link State). What's the other one called? Oh, dear. It's been so long since I worked on this stuff that I don't remember what the other ones are called. But anyway, I can explain it to you.

Oh, the hot potato?

Yeah, there's the hot potato stuff. But that's not what they are usually called these days. These things have changed names so many times over the last 25 years. I can't remember anymore. I'll explain briefly. In the first type, each IMP tried to figure out its best path to all the other IMPs as destinations and would report entire distance (vector). Okay? So the table that one IMP would send to all the other IMPs would be a list for all of the IMPs in the network. The distance from me, you know, one of them would be me, to all the other ones. Okay? The one that I

invented just said, *"Look. If here's me, I'm just going to report on the links that I know. So here are my links. A, B, C, and D."* In the first case, this IMP would calculate distances to everybody and then would exchange it only -- would send it only to neighbors. Then the neighbors would take these distances and the distances they got from other neighbors and calculate their answer. In the second case, you calculate only the traffic on the links that you know about directly, but then you send this to everybody.

This is Link State?

The Link State. And you send everywhere, which is sometimes called flooding. That means that this information about A, B, C, and D has to go to all of these places. That was thought in the early days of the network to be overly expensive, because, gee, you have to send it to everybody. But it turns out to have some really good mathematical properties that you can better optimal routes and you can get them more quickly. In the world of routers -- you know about routers with Cisco and _____ and all that -- all this stuff got reinvented. History repeats itself. It all got reinvented 15 years later. The routers use this as RIP routing information protocol. In the early days of the Arpanet -- I think we'll just call this the original Arpanet routing or whatever. This I gave the name shortest path first, and in the router world, they improved this to be called open shortest path first. SO even today in the mid-90's, you find that there are some routers that in simple networks still use RIP, which is the routing algorithm that I was replacing more than 20 years ago. And then of course, many of the other use (OSPF), which is a direct descendant of the thing that I developed in the mid-70's called SPF.

So in a general way, all this growth has come about is, as you say, following closely in the footsteps of the technology we developed. But even in a specific way, I mean, even the actual procedures -- now they've gotten improved and extended and so on. But basically, you saw this a lot, I'm sure, when you talked to Vint. TCP/IP was developed in 1974. It's 20 years old. It's gone through three major revisions, but it's still TCP/IP. And it's gone from supporting 40 computers to supporting two million computers, which is quite astounding really. It's astounding because no one who was building it was thinking about trying to make that one of their design goals. So in a way, it's kind of a happy accident that it works that well. It's not going to keep working that well for very much longer, as he probably told you.

It's not?

Well, depending on your point of view, IP is going to have to go through a fourth

change, and for some people, that's just one more little change, and for other people, it's a fairly major change. It's like if you keep calling it IP, if you call it IP Version 4, it sounds like it hasn't changed. IP Version 4 is going to have a totally new address venue. Totally new ways of coping with high performance traffic and it's going to be able to carry video and IP could never have carried video. So it's kind of like calling it still a Chevrolet even though the 1994 Chevrolet doesn't really look like the 1974 Chevrolet. It doesn't have the same engine or glass or air conditioning or emissions control or anything. But it is a Chevrolet, because each year it was only a small change, and it's the same guys who developed it. It's the same philosophy. In this case, it's a lot tighter analogy and a lot tighter following the technology. But then again, the numbers are very different because the network is very much faster, as well as very much bigger.

When you were young, what was your thinking? You knew you were in unchartered territory. But did you --

Yeah. I chose to do it for that reason.

It was really interesting to you.

Yeah, right. I mean, I was a grad student at Harvard and my first love was computer graphics and I would have really liked to have done that. In fact, I've kind of gone back to doing that now with video and animation and multi-media now that that's much more possible. And Arpa pulled a lot of the funding for graphics and started funding networking. I realized that if I wanted to be involved with cutting edge technology that networking was certainly going to be it. Ironically, at Harvard, there was no one on the faculty who was doing anything. So I went to BBN and I learned all this stuff. I developed the new algorithm and I wrote my PhD thesis describing this.

Oh, that was your PhD?

Yeah. And the faculty at Harvard were quite pleased that one of their students had done some original work. You know, as happens there and at other places, sometimes the original work doesn't happen entirely on campus. But it can. I wrote it on this huge thesis. But yeah, I quite deliberately wanted to do research and was looking for an interesting area that was going to be all new and BBN was where it was happening.

It was a good place to work.

It was a wonderful place to work. It was a wonderful place to work.

I've heard the culture in those days was wonderful.

Yeah. It was really, really wonderful. Especially for somebody like me who was interested in an intellectual challenge and who -- you know, I was very young and very bright and eager, and they just gave me a colossal amount of responsibility. I mean, I was just a kid and I was building all the software for the Arpanet. I really was, you know, personally, as I say, you know, putting all this software in week after week. It was just up to me to do it. My boss wasn't there watching me do it.

Who was your boss?

Frank Heart.

Oh, really?

Yeah. Frank was watching me in a general way, but he wasn't watching this happen. He just trusted me to do it. So that was very nice. And also, it had a very rich content of research in it. So I could invent things. Which is a very great pleasure in our business, because nowadays, everything has gotten so bogged down with so many people doing so much, that there's very little scope for inventions. There's standards, and there's committees, and there's rules, and there's a lot of established industry practice. When I wanted to develop a new way of doing something, I'd start with a clean sheet of paper and I'd start drawing what I wanted the message to look like. *"Okay. I'll put the ack here, and I'll put the source here, and I'll put the destination here."* And it was just up to me. You know, I'd talk about it with a few other BBN people. I don't mean to be egotistical. I mean it was --

Much easier.

Everybody else at BBN was in the same position. The people who were developing hardware. They were all just able to develop it the way they wanted. We would collaborate with each other and then we would implement it. That's a tremendous luxury. And that's what I was seeking, because that's what you get in the university environment. The distinction with computers as opposed to some other academic disciplines is that it's not enough to do it on paper. You have to implement it to know if it's going to be valuable and that's why BBN was perfect, because it was a combination of the opportunities you get in the academic world, plus the reality of the business world.

Exactly. I've heard that people loved it because you got to do all that research. You didn't have to teach.

Right.

And the environment. I've spent a lot of time over there doing the book. A lot of those guys are still there.

Yeah, they are. It's amazing. It really is. I worked for Frank for 10 years. I like to say that I've only had one job interview. I went to interview with Frank after getting my Master's Degree. He hired me and I worked there for 10 years and I just left to go out on my own. It sort of spoiled me from doing anything else. I'm not going to ever have a real job. I just did that and now I work for myself, because that's about the only other thing I could think of that would be as free as that.

You have to stop, don't you?

I can go until my call comes. But it is 4:00. Did we get to the nitty gritty of what you wanted?

Yeah, this is good. Can I take that?

Yeah, sure. I was drawing this so you could take this.

Obviously, I'm going to want you to read anything I write. What was it like working for Frank?

Frank was a tremendous boss and is a tremendous boss. I have a very, very high regard for him.

Yeah. I just think he's great.

He's a really unusual person. A real unusual person.

What makes him unusual?

A lot of personal integrity. He really never says anything that he doesn't believe in. So he just built systems that he believed in. He built them the way they should be built and he didn't want to compromise any -- he wasn't interested in having people work for him who were either phonies or who weren't able to do really good work.

So he was demanding, but he was a really good boss. I think the thing that I appreciated a lot in the early days, at least, was just that he let me do as much as I was capable of doing. So I got some really, really hard jobs to do. Took me a long time, but I did them. The reward, you know, was to get more. More of those. So that was very nice.

I also think that Frank is somebody with a very good people sense. For a technology guy, he is surprisingly a people guy. You might not even think it at first, because he's really involved with technology and a lot of technologists are just a zero in social skills and also in sensitivity to people. And Frank's been able to manage just hundreds of people over the years of all kinds. He's been a manager for decades and I think he's just really good at figuring out, *"Okay, well, this person's really good at this and that person's really good at that. So let's have then do these different things."* In the business world, it's much more common to find people who are pretty callous or who don't really have sensitivity about _____ things.

Then finally, Frank's had very good technical judgment and technical vision all these years, which is also quite impressive. Because a lot of people know what they know and don't know what they don't know. I think Frank has been right about a lot of things. So I have a lot of regard for him.

Yeah. I just spent the weekend up in Maine with him.

Oh, did you?

Yes.

[Talk about the weekend.]

He would be the first to say he thinks he has a lot of social awkwardness or he thinks he never really got A's in personality.

But it's just manner. Where it really counts, he's tremendous. Then there are managers quite often who are the reverse. Who are Mr. Smooth and who are very polished and everything, but they're just cutting people off at the knees all the time. Frank would never do that. He's not a nasty person. You know, even though he seems kind of -- I don't know -- energetic and forceful and everything. But he's not a nasty guy. Whereas, the business world is filled with nasty guys. They're just completely heartless S.O.B.'s.

I know. And it's been much to Frank's dismay over the years seeing some of these guys come in and out of BBN.

Yeah.

He take loyalty to heart. When people leave, he take it hard.

Yeah. He's got his -- I mean, the flip side of these things. He invests a lot of himself in these relationships and so, yeah. Well, he (brings) his heart in his sleeve. He's just a very emotional guy.

[Tape is turned off.]

Infinitely faster and infinitely bigger. I think it might make a nice little comment on the history of technology to say that the stuff is evergreen. It just keeps going on.

And time (both) to people. You know, Larry Roberts --

Larry Roberts is just doing this right now.

Have you talked to him much?

Of course, yeah. I talk with him. Sure. Absolutely. The IMPs were doing 56 kilobit lines. ATM switches to 150 megabit lines. So it's 3000 times faster.

That's amazing.

3000 times. And all of the same questions of how you do routing algorithms, and how you do acknowledgements, and how you do congestion control, and everything, are all around. But now we're talking about building networks with millions of points and with billions of bits a second, instead of thousands of computers and tens of kilobits a second. So it's a very weird position to be in, in a way. I feel like I've kind of stayed at one point in technology and I've been watching the river flow by me. You know?

Uh-huh.

And each year the river just gets wider and wider. Sometimes I feel like I'm stuck and other times I feel like I'm getting blown away and carried away because so much is going on. But I do think that just reminding people of historical context here is

very important. Because this is true for science, too. I mean, you have like Crick figuring out about DNA and 25 years later you have people engineering this stuff. Just with most science and with engineering and technology, I think that's a very interesting part of the story. There's the first time and then there's the what happens to it after it gets really, really big.

Well, and the fact that all you guys are still very much -- well, you say you're getting more into graphics.

Well, that's just because what you can do is once you have these kinds of networks you can send video over them. Whereas, with the Arpanet, we were happy to send text. So yeah, but that's just to keep me from going crazy that it's the same stuff all the time. I am still very much involved in networking. I just meant that it's giving me a chance to go back to my first love, which is to do some graphic and now video things with computers that just weren't possible before.

That's a wonderful point. I should use that and Larry as the book ends.

Yeah. Just kind of go back. You say, *"Okay, here's Larry's office in 1968 and here's Larry today. What's he doing?"* He's building a multi-gigabit ATM switch for like $3.00 and it's this big, and the old IMPs were huge and very expensive.

That's a great idea.

Yeah, and in fact, I think many of the other people you might have been talking to are also involved. Certainly, Vint with the Internet Society.

And Bob Kahn.

Bob Kahn with the gigabit test bits and everything. That's right. You could almost say that almost everyone in the inner circle is still involved in one way or another.

Good. Thanks a lot.

Sure.

[End of interview.]